



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/893,645	06/29/2001	John Charles Brook	169.2089	5805

5514 7590 12/22/2006
FITZPATRICK CELLA HARPER & SCINTO
30 ROCKEFELLER PLAZA
NEW YORK, NY 10112

EXAMINER

TRAN, QUOC A

ART UNIT	PAPER NUMBER
----------	--------------

2176

SHORTENED STATUTORY PERIOD OF RESPONSE	MAIL DATE	DELIVERY MODE
3 MONTHS	12/22/2006	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

If NO period for reply is specified above, the maximum statutory period will apply and will expire 6 MONTHS from the mailing date of this communication.

Office Action Summary	Application No. 09/893,645	Applicant(s) BROOK, JOHN CHARLES	
	Examiner Tran A. Quoc	Art Unit 2176	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

- 1) ☒ Responsive to communication(s) filed on 06 October 2006.
- 2a) ☒ This action is **FINAL**. 2b) ☐ This action is non-final.
- 3) ☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

- 4) ☒ Claim(s) 1-7, 9-17, 20-26, 29-31, 34, 39, 44, 47, 48 and 59-62 is/are pending in the application.
- 4a) Of the above claim(s) _____ is/are withdrawn from consideration.
- 5) ☐ Claim(s) _____ is/are allowed.
- 6) ☒ Claim(s) 1-7, 9-17, 20-26, 29-31, 34, 39, 44, 47-48, and 59-62 is/are rejected.
- 7) ☐ Claim(s) _____ is/are objected to.
- 8) ☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

- 9) ☐ The specification is objected to by the Examiner.
- 10) ☐ The drawing(s) filed on _____ is/are: a) ☐ accepted or b) ☐ objected to by the Examiner.
Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).
Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
- 11) ☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

- 12) ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
- a) ☐ All b) ☐ Some * c) ☐ None of:
1. ☐ Certified copies of the priority documents have been received.
2. ☐ Certified copies of the priority documents have been received in Application No. _____.
3. ☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).
- * See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

- | | |
|--|---|
| 1) <input checked="" type="checkbox"/> Notice of References Cited (PTO-892) | 4) <input type="checkbox"/> Interview Summary (PTO-413)
Paper No(s)/Mail Date. _____ |
| 2) <input type="checkbox"/> Notice of Draftsperson's Patent Drawing Review (PTO-948) | 5) <input type="checkbox"/> Notice of Informal Patent Application |
| 3) <input type="checkbox"/> Information Disclosure Statement(s) (PTO/SB/08)
Paper No(s)/Mail Date _____ | 6) <input type="checkbox"/> Other: _____ |

DETAILED ACTION

1. This action is a final rejection in response to amendment filed on 10/06/2006.
2. Claims 1-7, 9-17, 20-26, 29-31, 34, 39, 44, 47-48 and 59-62 are pending.
3. Claims 1, 6-7, 9-11, 14, 16, 17, 21-26, 29, 34, 39, 44, 47 and 48 have been amended.
4. Claims 59-62 are newly added.
5. Effective filing date 07/16/2004, benefits from US Provisional 60/148,172 filed 8/10/1999.

Claim Objection

6. Claims 2, 5, 12, 15, and 30-31 (see claims pages 2-3, 5, 7 and 10-11) objected to because of the following informalities. Claims 2, 5, 12, 15, and 30-31 of amendment filed 10/06/2006 does not provide the correct status identifier. Appropriate correction is required (see MPEP 37 CFR 1.121),

(A) Status Identifiers: The current status of all of the claims in the application, including any previously canceled or withdrawn claims, must be given. Status is indicated in a parenthetical expression following the claim number by one of the following status identifiers: (original), (currently amended), (previously presented), (canceled), (withdrawn), (new), or (not entered).

Claim Rejections - 35 USC § 103

7. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

Claims 1-7, 9-17, 20-26, 29-31, 34, 39, 44, 47-48 and 59-62, are rejected under 35 U.S.C. 103(a) as being unpatentable by Hind et al. US006635088B1- filed 11/20/1998 (hereinafter Hind), in view of Girardot et al. US006883137B1 filed 04/17/2000 (hereinafter Girardot).

Regarding independent claim 1, Hind teaches a method of generating encoded representation of a markup language document comprising syntactic elements, said method comprising, the steps of: parsing the markup language document to identify at least one syntactical element of that document. For example, Hind discloses a compression of a structured document and a document type definition (Hind, Title), where XML and DTD files are compressed, reducing the file size for storage and transfer (Hind, col 3, lines 62-67). Specifically, an encoded file is read and each of the tags is located in the encoded file. Thereafter, a subprocess substitutes a unique short tag for each of the unique one of the located tags (Hind, col 4, lines 25-30). In addition, Hind discloses the XML parsers is utilized in the manner in which such parsers are designed to process "entities" within a file (an entity is "a collection of characters that can be referenced as a unit", according to the SGML standard), wherein an entity is encoded using a special XML constructs, and is used to substitute a text string or a file into a document. For purposes of the present invention, the pertinent substitution is the text string mechanism). The examiner interprets the claimed **encoded representation of a markup language document comprising syntactic elements of the claimed** as equivalent to Hind's XML parsers processing "entities" within a file to substitute a text string or a file into a document, and because the Applicant Specification defines, " this idea allows an arbitrary XML tag to be treated as a numeral or code" (Applicant specification page 7 para 207).

In addition, Hind teaches **generating the encoded representation including the numeric code if the type is the first type**. Specifically, Hind discloses

<!ENTITY entity_name "the text string for the entity">

or entity declaration: <!ENTITY A "B">, and entity reference: &A, declares an entity named "A", and substitution text containing the single character "B". Thus, when the symbol "&A;" is detected in the file by the parser, the character "B" will be automatically inserted in its place. An entity declaration will be created for the strings that appear in a document file being compressed, replacing the strings with a shorter entity reference, thereby compressing the amount of space required for the string (the decompression is performed by the XML parser) (col 8, lines 59-65; col 9, lines 20-35). The Examiner equates the claimed **representation including the numeric code** to Hind's teaching of entity declaration: a shorter entity reference, wherein (an entity is "a collection of characters that can be referenced as a unit").

Hind discloses the logic for processing of substitution of references between tags (Hind, col 10, lines 27-40). Hind does not teach, but Girardot teaches **identifying a type of the element**. Specifically discloses element can contained character data, child elements or a mixed of both (Girardot, col 10, lines 27-40).

(line 1)	<?xml version="1.0"?>
(line 2)	<Book Author="John Steinbeck" Genre="literature">
(line 3)	<Title>Of Mice and Men</Title>
(line 4)	<Chapter id="1">
(line 5)	Blah . . . Blah . . .
(line 6)	<Chapter id="2">
(line 7)	Blah . . . Blah . . .

(line 8) </Chapter>

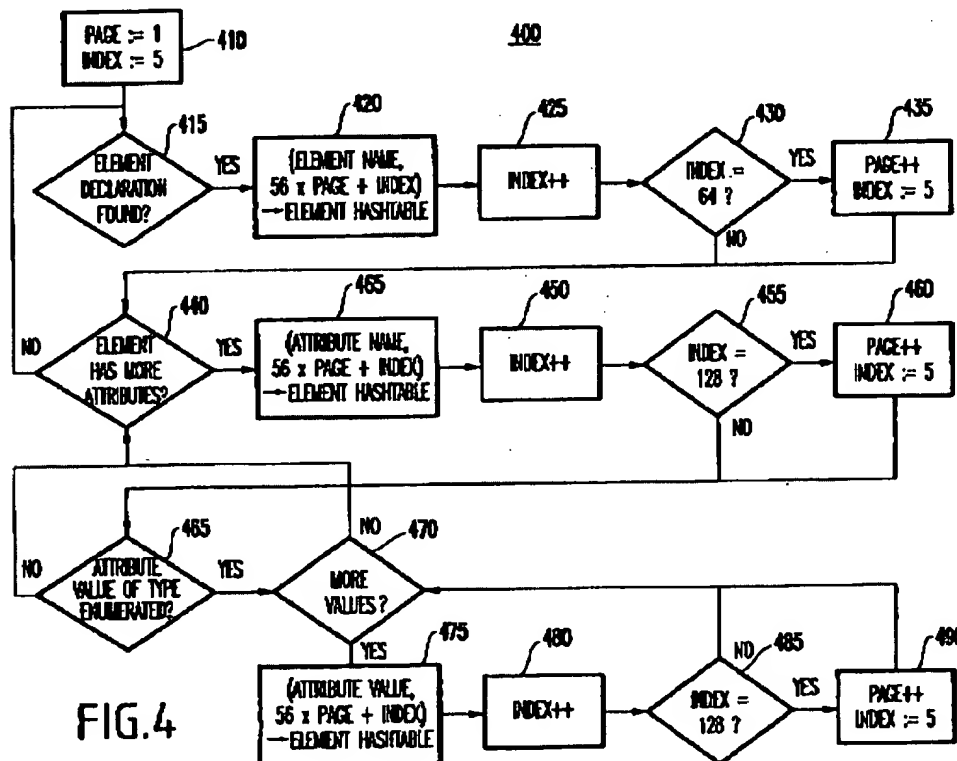
(line 9) </Book>

The Examiner equates the claimed **element type** as equivalent to Hind's teaching of element can contained character data, child elements or a mixed of both.

Hind does not teach, but Girardot teaches **processing the element by applying a hash function thereto if the type is a first type, the hash function generating a numeric code from the element**. Specifically Girardot discloses a hash table is used on the server for a specific tag name or attribute name or value (e.g., which are of string type), and then proceed to step 320 parsing a Document Type Definition (DTD) produces a tree where elements and attributes can be accessed by name (Girardot, col 7, lines 45-65). Furthermore, Girardot discloses server code spaces is performed by the server filling in the hash table for an element code space. That is, in step 410, the page number variable is set to 0, and the index variable to 5 (e.g., in the exemplary implementation the first four indexes are reserved for global tokens) (Girardot, col 7, lines 45-65, fig. 3-4). The examiner interprets the claimed **the type is a first type** as equivalent to Girardot's teaching of string type, and the claimed **hash function generating a numeric code** as equivalent to Girardot's teaching of hash table is used on the server for a specific tag name or attribute name or value wherein server code spaces is performed by the server filling in the hash table for an element code space, and because the Applicant Specification defines, " this idea allows an arbitrary XML tag to be treated as a numeral or code" (Applicant specification page 7 para 207).

TAG CODE SPACE		ATTRIBUTE NAME CODE SPACE		ATTRIBUTE NAME CODE SPACE	
TAG NAME	TOKEN	ATTRIBUTE NAME	TOKEN	ATTRIBUTE VALUE	TOKEN
BOOK	5	AUTHOR	5	LITERATURE	5
TITLE	6	GENRE	6	SCIENCE	6
CHAPTER	7	NUMBER	7	HISTORY	7
PICTURE	8	CAPTION	8	CARTOONS	8

HASH TABLE FOR THE CODE SPACES ON THE SERVER SIDE



It would have been obvious to one of ordinary skill in the art at the time the invention to modify the teaching of Hind to include identifying a type of the element, and processing the element by applying a hash function thereto if the type is a first type as taught by Girardot, because Hind, and Girardot are from the same field of endeavor of providing a method, and

system which utilizing XML Syntactical properties to compress and validating XML element data, and to minimizing the size of files being transmitted and stored (Hind col 3, lines 5-30).

Regarding independent claims 34, 39, 44, and 47, the rejection of claim 1 is fully incorporated.

Regarding independent claim 48, the rejection of claim 1 is fully incorporated. In addition Hind teaches **a processor, a memory for storing (i) the document, and (ii) a program, which is configured to make the processor execute a procedure to generate the encoded representation**. Specifically Hind discloses a compression of a structured document and a document type definition (Hind, Title), where XML and DTD files are compressed, reducing the file size for storage and transfer (Hind, col 3, lines 62-67). Specifically, an encoded file is read and each of the tags is located in the encoded file. Thereafter, a subprocess substitutes a unique short tag for each of the unique one of the located tags (Hind, col 4, lines 25-30).

Regarding claim 2, Hind teaches where **parsing is event-based parsing**. The examiner interprets the claimed event based parsing as parsing to provide lower-level access to XML document, facilitating parsing of documents larger than available system memory by parsing start and end of elements (see Applicant's specification, pages 1 and 2). Hind discloses opening and matching ending tags (Hind col 2, lines 25-30) and a need for reducing the length of tags for processing at a constrained-storage device (col 3, lines 1-10), where the parser reads each entity declaration in sequence (Hind col 9, lines 5-8).

Regarding claim 3, Hind does not expressly teach, but Girardot teaches a method, **wherein said the numeric code is determined using one of: a hash algorithm; a first**

reference to the hash algorithm dependent upon an associated Universal Reference Indicator; a second reference to said the hash algorithm dependent upon an associated namespace; and the reference to said the hash algorithm dependent upon an associated Extended Markup Language declaration. For example Girardot teaches **identifying a type of the element**. Specifically Girardot discloses a hash table is used on the server for a specific tag name or attribute name or value (e.g., which are of string type), and then proceed to step 320 parsing a Document Type Definition (DTD) produces a tree where elements and attributes can be accessed by name (Girardot, col 7, lines 45-65). Furthermore, Girardot discloses XML Validating parser to ensure that the documents conformed to the language (Girardot col. 4, lines 60-65). Furthermore Girardot discloses hereinbelow some exemplary code of the present invention, wherein system to fill in this element code space. That is, for each element declaration, the system gets the element name from the array on the client side, and the hash table on the server side, and the required and fixed attributes from the array on the client side, and the hash table on the server side (Girardot, col 10, lines 40-35): `*@param dtd a <code>DTD</code> object */`. The examiner interprets the claimed **a hash algorithm** as equivalent to Girardot's teaching of hash table, and the claimed **the syntactic rule relates to proper nesting of tags** as equivalent to Girardot's teaching of `*@param dtd a <code>DTD</code> object */`, and the claimed **hash algorithm dependent upon an associated namespace** as equivalent to Girardot's teaching of `*@param dtd a <code>DTD</code> object*/`.

It would have been obvious to one of ordinary skill in the art at the time the invention to modify the teaching of Hind to include wherein said the numeric code is determined using one of: a hash algorithm; a first reference to the hash algorithm dependent upon an associated

Art Unit: 2176

Universal Reference Indicator; a second reference to said the hash algorithm dependent upon an associated namespace; and the reference to said the hash algorithm dependent upon an associated Extended Markup Language declaration as taught by Girardot, because Hind, and Girardot are from the same field of endeavor of providing a method, and system which utilizing XML Syntactical properties to compress and validating XML element data, and to minimizing the size of files being transmitted and stored (Hind col 3, lines 5-30).

Regarding claim 4, Hind teaches **wherein said first type is a declaration of the structural element**. Hind discloses declaration of an entity (Hind col 8, lines 59-65; col 9, lines 20-35 shows an example of an entity declaration).

Regarding claim 5, Hind teaches **wherein the structural element is a tag**. Hind discloses a file encoded according XML element tag (Hind col 8, lines 55-59).

Regarding claim 6, Hind teaches **the element having less than a first number of characters**. For example Hind discloses an entity declaration will be created for the strings that appear in a document file being compressed, replacing the strings with a shorter entity reference, thereby compressing the amount of space required for the string (the decompression is performed by the XML parser) (col 8, lines 59-65; col 9, lines 20-35).

Hind does not teach, but Girardot teaches **numeric code is a unique code for the one syntactic element, the element having less than a first number of characters**. Specifically Girardot discloses a hash table is used on the server for a specific tag name or attribute name or value (e.g., which are of string type), and then proceed to step 320 parsing a Document Type Definition (DTD) produces a tree where elements and attributes can be accessed by name

Art Unit: 2176

(Girardot, col 7, lines 45-65). The Examiner equates the claimed **numeric code is a unique code** as equivalent to a hash table value.

It would have been obvious to one of ordinary skill in the art at the time the invention to modify the teaching of Hind to include numeric code is a unique code for the one syntactic element, the element having less than a first number of characters as taught by Girardot, because Hind, and Girardot are from the same field of endeavor of providing a method, and system which utilizing XML Syntactical properties to compress and validating XML element data, and to minimizing the size of files being transmitted and stored (Hind col 3, lines 5-30).

Regarding claim 7, Hind does not teach, but Girardot teaches **numeric code is not a unique code for the one syntactic element, the element being constrained, to a probability level, in term of a number of characters in the element**. For example, Girardot discloses XML Validating parser to ensure that the documents conformed to the language (Girardot col. 4, lines 60-65). Furthermore Girardot discloses a SAX event 270 for each received element. Possibly, the system can also build dynamically a DOM tree 280. Each time an element is received, it is added dynamically to the DOM tree (Girardot col. 7, lined 25-30, Fig. 2).

It would have been obvious to one of ordinary skill in the art at the time the invention to modify the teaching of Hind to include numeric code is not a unique code for the one syntactic element, the element being constrained, to a probability level, in term of a number of characters in the element as taught by Girardot, because Hind, and Girardot are from the same field of endeavor of providing a method, and system which utilizing XML Syntactical properties to compress and validating XML element data, and to minimizing the size of files being transmitted and stored (Hind col 3, lines 5-30).

Regarding claim 9, the rejection of claim 1 is fully incorporated. In addition Hind teaches a sub-step of: **(i) the one syntactic element being a first instance of the first type, and (ii) another syntactic element being a second instance of the first type, within which first instance, said the second instance is nested.** Hind discloses, in Fig 5B, a block 505' with tags "<A>" containing nested tags of type "", "<C>", "<D>" and "<E>" which map to longer tag values seen in Fig 5A, block 505. Figures 5A and 5B show a nested structure where Customer_Nbr, Customer_Name and Ship_To_Address are nested within the block of Order.

Regarding claim 10, the rejection of claim 1 is fully incorporated. In addition **identifying a type of the other element; and if the type of the other element is equivalent to said first type: (i) processing the other element by apply the hash function thereto, thereby to generate a numeric code from the other element and (ii) augmenting the enclosed representation of the document using the second hash representation of the markup language document using the second numeric code, wherein: said processing of the one element and said second processing of the other element ensure that if a first relationship exists between the one element and the other element, then a second relationship which is representative of the first relationship, exists between the numeric code of the one element and the second numeric code of the other element.** Specifically, Girardot discloses for each element declaration (step 415) the server system gets the element name, adds it in the hash table with the element name as the key and $(256 \times \text{pageNumber} + \text{index})$ as the value (step 420), (Girardot, col 8, lines 15-30). In addition, Girardot discloses the names of the elements and attributes in the hierarchy form the XML markup language used by the document. In the example shown below, the language contains three elements: Book, Title, and Chapter. The Book element

Art Unit: 2176

contains a single Title element and one or more Chapter elements. The Book element has an Author attribute and a Genre attribute and the Chapter element has an id attribute (e.g., "2").

(Girardot, col 4, lines 30- 60, fig. 3-4):

```
(line 1)      <?xml version="1.0"?>
(line 2)      <Book Author="John Steinbeck" Genre="literature">
(line 3)          <Title>Of Mice and Men</Title>
(line 4)          <Chapter id="1">
(line 5)              Blah . . . Blah . . .
(line 6)          <Chapter id="2">
(line 7)              Blah . . . Blah . . .
(line 8)          </Chapter>
(line 9)      </Book>
```

The examiner interprets the claimed as equivalent to Girardot's teaching of hash table is used on the server for a specific tag name or attribute name or value wherein server code spaces is performed by the server filling in the hash table for an element code space, and because the Applicant Specification defines, " this idea allows an arbitrary XML tag to be treated as a numeral or code" (Applicant specification page 7 para 207).

It would have been obvious to one of ordinary skill in the art at the time the invention to modify the teaching of Hind to include identifying a type of the other element; and if the type of the other element is equivalent to said first type: (i) processing the other element by apply the hash function thereto, thereby to generate a numeric code from the other element and (ii) augmenting the enclosed representation of the document using the second hash representation of the markup language document using the second numeric code, wherein: said processing of the one element and said second processing of the other element ensure that if a first relationship

Art Unit: 2176

exists between the one element and the other element, then a second relationship which is representative of the first relationship, exists between the numeric code of the one element and the second numeric code of the other element as taught by Girardot, because Hind, and Girardot are from the same field of endeavor of providing a method, and system which utilizing XML Syntactical properties to compress and validating XML element data, and to minimizing the size of files being transmitted and stored (Hind col 3, lines 5-30).

Regarding claim 11, Hind teaches where one element is a start tag; the other element is an end tag. For example Hind discloses opening tag and a matching closing (or “end”) tag (Hind col 2, lines 26-27). Fig 5A, items 560 and 561, show opening and closing tags, respectively. Fig 5B, items 560’ and 561’, show opening and closing tags after compression (Hind col 13, lines 9 –15).

Hind does not teach, but Girardot teaches **the numeric code of the one element is a corresponding hashed start tag, the second numeric code of the other element is a corresponding hashed end tag.** For example Girardot discloses hereinbelow some exemplary code of the present invention, wherein system to fill in this element code space. That is, for each element declaration, the system gets the element name from the array on the client side, and the hash table on the server side, and the required and fixed attributes from the array on the client side, and the hash table on the server side (Girardot, col 10, lines 40-35): `*@param dtd a <code>DTD</code> object */`. The examiner interprets the claimed as equivalent to Girardot’s teaching of hash table is used on the server for a specific tag name or attribute name or value wherein server code spaces is performed by the server filling in the hash table for an element code space, for example `*@param dtd` is parameter element to a specific element type,

`<code>DTD</code>` is start tag and end tag in code instead of the input string of a hash table, and because the Applicant Specification defines, “ this idea allows an arbitrary XML tag to be treated as a numeral or code” (Applicant specification page 7 para 207).

It would have been obvious to one of ordinary skill in the art at the time the invention to modify the teaching of Hind to include the hash representation of the one element is a corresponding hashed start tag, the numeric code of the other element is a corresponding hashed end tag as taught by Girardot, because Hind, and Girardot are from the same field of endeavor of providing a method, and system which utilizing XML Syntactical properties to compress and validating XML element data, and to minimizing the size of files being transmitted and stored (Hind col 3, lines 5-30).

Regarding claim 12, the rejection of claim 11 is fully incorporated. In addition Hind teaches **where the end tag is a first modification of the start tag; and the hashed end tag is a second modification of the hashed start tag, said second modification being representative of the first modification**. Hind discloses opening tag and a matching closing (or “end”) tag (col 2, lines 26-27). Fig 5A, items 560 and 561, show opening and closing tags, respectively. Fig 5B, items 560' and 561', show opening and closing tags after compression (col 13, lines 9 –15). The closing tags include a “/” symbol with an open bracket angle that designates that it is the end tag (col 2, lines 30-35).

Regarding claim 13, the rejection of claims 11-12 is fully incorporated. In addition Hind teaches **where the end tag is the same as the start tag apart from having a distinguishing character incorporated therein; and the hashed end tag is the same as the hashed start tag apart from having a distinguishing character incorporated therein**. Hind discloses opening

Art Unit: 2176

tag and a matching closing (or “end”) tag (col 2, lines 26-27). Fig 5A, items 560 and 561, show opening and closing tags, respectively. Fig 5B, items 560' and 561', show opening and closing tags after compression (col 13, lines 9 –15). The closing tags include a “/” symbol with an open bracket angle that designates that it is the end tag (col 2, lines 30-35). Fig 5A shows an example where the start tag is “<order>” and the matching end tag is “</order>”, which is compressed in Fig 5B as “<A>” and the matching end tag is “”.

Regarding claim 14, the rejection of claims 11-12 is fully incorporated. In addition Hind does not teach, but Girardot teaches **the document further includes a second pair of tags comprising a respective start tag and end tag, said second pair of tags being nested within said first pair of tags in said document, said method comprising further steps of: processing said second pair of tags to form corresponding second hashed start and end tags.** For example Girardot discloses hereinbelow some exemplary code of the present invention, wherein system to fill in this element code space. That is, for each element declaration, the system gets the element name from the array on the client side, and the hash table on the server side, and the required and fixed attributes from the array on the client side, and the hash table on the server side (Girardot, col 10, lines 40-35): `*@param dtd a <code>DTD</code> object */`. The examiner interprets the claimed as equivalent to Girardot's teaching of hash table is used on the server for a specific tag name or attribute name or value wherein server code spaces is performed by the server filling in the hash table for an element code space, for example `*@param dtd` is parameter element to a specific element type, `<code>DTD</code>` is start tag and end tag in code instead of the input string of a hash table, and because the Applicant Specification defines, “ this idea

Art Unit: 2176

allows an arbitrary XML tag to be treated as a numeral or code" (Applicant specification page 7 para 207).

Furthermore, Hind does not teach, but Girardot teaches Girardot teaches **augmenting said at least partial structural representation of the document using said corresponding second hashed start and end tags so that said second hashed start and end tags indicate a nesting in relation to said hashed start and end tags for the first pair of tags which is equivalent to the nesting of said second pair of tags within said first pair of tags.**

Specifically Girardot discloses hereinbelow some exemplary code of the present invention, wherein system to fill in this element code space. That is, for each element declaration, the system gets the element name from the array on the client side, and the hash table on the server side, and the required and fixed attributes from the array on the client side, and the hash table on the server side (Girardot, col 10, lines 40-35): `*@param dtd a <code>DTD</code> object */`. The examiner interprets the claimed **start and end tags indicate a nesting in relation to said hashed start and end tags** as equivalent to Girardot's teaching of hash table is used on the server for a specific tag name or attribute name or value wherein server code spaces is performed by the server filling in the hash table for an element code space, for example `*@param dtd` is parameter element to a specific element type, `<code>DTD</code>` is start tag and end tag in code instead of the input string of a hash table, and because the Applicant Specification defines, "this idea allows an arbitrary XML tag to be treated as a numeral or code" (Applicant specification page 7 para 207).

It would have been obvious to one of ordinary skill in the art at the time the invention to modify the teaching of Hind to include the hash representation of the one element is a

corresponding hashed start tag and end tag, the numeric code of the other element is a corresponding hashed end tag as taught by Girardot, because Hind, and Girardot are from the same field of endeavor of providing a method, and system which utilizing XML Syntactical properties to compress and validating XML element data, and to minimizing the size of files being transmitted and stored (Hind col 3, lines 5-30).

Regarding claim 15, the rejection of claim 14 is fully incorporated. In addition Hind does not teach, but Girardot teach **a further step of concatenating the first hashed start tag with the second hashed start tag, and concatenating the first hashed end tag with the second hashed end tag, to thereby form respective extended hashed start and end tags for said second pair, wherein: said augmenting step is performed using said respective extended hashed start and end tags for said second pair, and; said extended hashed start and end tags indicate a nesting in relation to said hashed start and end tags for the first pair of tags which is equivalent to the nesting of said second pair of tags within said first pair of tags**. Specifically Girardot discloses hereinbelow some exemplary code of the present invention, wherein system to fill in this element code space. That is, for each element declaration, the system gets the element name from the array on the client side, and the hash table on the server side, and the required and fixed attributes from the array on the client side, and the hash table on the server side (Girardot, col 10, lines 40-35): `*@param dtd a <code>DTD</code> object */`. The examiner interprets the claimed **concatenating the first hashed start tag with the second hashed start tag** as equivalent to Girardot's teaching of hash table is used on the server for a specific tag name or attribute name or value wherein server code spaces is performed by the server filling in the hash table for an element code space, for example

*@param dtd is parameter element to a specific element type, `<code>DTD</code>` is start tag and end tag in code instead of the input string of a hash table, and because the Applicant Specification defines, “ this idea allows an arbitrary XML tag to be treated as a numeral or code” (Applicant specification page 7 para 207).

It would have been obvious to one of ordinary skill in the art at the time the invention to modify the teaching of Hind to include concatenating the first hashed start tag with the second hashed start tag, and concatenating the first hashed end tag with the second hashed end tag, to thereby form respective extended hashed start and end tags for said second pair, wherein: said augmenting step is performed using said respective extended hashed start and end tags for said second pair, and; said extended hashed start and end tags indicate a nesting in relation to said hashed start and end tags for the first pair of tags which is equivalent to the nesting of said second pair of tags within said first pair of tags as taught by Girardot, because Hind, and Girardot are from the same field of endeavor of providing a method, and system which utilizing XML Syntactical properties to compress and validating XML element data, and to minimizing the size of files being transmitted and stored (Hind col 3, lines 5-30).

Regarding claim 16, the rejection of claim 1 is fully incorporated. In addition Hind do not expressly teach, but Girardot teaches wherein **said augmenting step is succeeded by a well-formedness checking step against a syntactic rule, said well-formedness checking step comprising checking the markup language document against the syntactic rule by numerically comparing corresponding numeric code of elements in representation of the markup language document**. For example, Girardot discloses XML Validating parser to ensure that the documents conformed to the language (Girardot col. 4, lines 60-65). Furthermore

Art Unit: 2176

Girardot discloses a SAX event 270 for each received element. Possibly, the system can also build dynamically a DOM tree 280. Each time an element is received, it is added dynamically to the DOM tree (Girardot col. 7, lined 25-30, Fig. 2). The examiner interprets the claimed **well-formedness checking step** as equivalent to Girardot's teaching of XML Validating parser to ensure that the documents conformed to the language of SAX event, and because and because the Applicant Specification defines, "FIGS. 2(a) and 2(b) depict a prior art SAX parser process 236, which supports optional well-formedness and/or validation checking sub-processes" (Applicant specification page 7 para 209, fig. 2a-b).

It would have been obvious to one of ordinary skill in the art at the time of the invention to modify Hind and Girardot to include well-formedness as taught by Girardot, providing the benefit of ensuring that the documents conformed to the (Girardot col. 4, lines 60-65).

Regarding claim 17, Hind teaches **wherein said numerically comparing step is succeeded by a further step of string-comparing, in accordance with the syntactic rule, corresponding non-process representations of elements not of the first type**. Hind discloses entries in a table of strings where once all the string have been located and the strings that meet the condition have been entered into the string table, the logic processes this string table, using it to perform the substitution of entries for strings within the input file (col 10, line 66 – col 11, line 15).

Regarding claim 20, Hind does not expressly teach, but Girardot teaches a method, comprising a further step of: **checking the well-formedness of the encoded representation of the document against a syntactic rule**. For example, Girardot discloses XML Validating parser to ensure that the documents conformed to the language (Girardot col. 4, lines 60-65).

Furthermore Girardot discloses a SAX event 270 for each received element. Possibly, the system can also build dynamically a DOM tree 280. Each time an element is received, it is added dynamically to the DOM tree (Girardot col. 7, lined 25-30, Fig. 2). The examiner interprets the claimed **well-formedness checking step** as equivalent to Girardot's teaching of XML Validating parser to ensure that the documents conformed to the language of SAX event, and because and because the Applicant Specification defines, "FIGS. 2(a) and 2(b) depict a prior art SAX parser process 236, which supports optional well-formedness and/or validation checking sub-processes" (Applicant specification page 7 para 209, fig. 2a-b).

It would have been obvious to one of ordinary skill in the art at the time of the invention to modify Hind to include well-formedness as taught by Girardot, providing the benefit of ensuring that the documents conformed to the (Girardot col. 4, lines 60-65).

Regarding claim 21, Hind does not expressly teach, but Girardot teaches a method, comprising a further step of: **wherein the syntactic rule relates to proper nesting of tags and said checking step comprises sub-steps of: performing a numerical comparison across hashed tags in said at least partial structural representation of the document to thereby identify said first hashed start and end tags and said second hashed start and end tags; and verifying that the second hashed start and end tags indicate a proper nesting in relation to said first hashed start and end tags.** For example, Girardot discloses XML Validating parser to ensure that the documents conformed to the language (Girardot col. 4, lines 60-65). Furthermore Girardot discloses a SAX event 270 for each received element. Possibly, the system can also build dynamically a DOM tree 280. Each time an element is received, it is added dynamically to the DOM tree (Girardot col. 7, lined 25-30, Fig. 2). Also Girardot discloses hereinbelow some

Art Unit: 2176

exemplary code of the present invention, wherein system to fill in this element code space. That is, for each element declaration, the system gets the element name from the array on the client side, and the hash table on the server side, and the required and fixed attributes from the array on the client side, and the hash table on the server side (Girardot, col 10, lines 40-35): `*@param dtd` a `<code>DTD</code>` object `*/`. The examiner interprets the claimed **first hashed start and end tags** as equivalent to Girardot's teaching of hash table is used on the server for a specific tag name or attribute name or value wherein server code spaces is performed by the server filling in the hash table for an element code space, for example `*@param dtd` is parameter element to a specific element type, `<code>DTD</code>` is start tag and end tag in code instead of the input string of a hash table, and because the Applicant Specification defines, "this idea allows an arbitrary XML tag to be treated as a numeral or code" (Applicant specification page 7 para 207), and the claimed **the syntactic rule relates to proper nesting of tags** as equivalent to Girardot's teaching of XML Validating parser to ensure that the documents conformed to the language of SAX event, and because and because the Applicant Specification defines, "FIGS. 2(a) and 2(b) depict a prior art SAX parser process 236, which supports optional well-formedness and/or validation checking sub-processes" (Applicant specification page 7 para 209, fig. 2a-b).

It would have been obvious to one of ordinary skill in the art at the time the invention to modify the teaching of Hind to include wherein the syntactic rule relates to proper nesting of tags and said checking step comprises sub-steps of: performing a numerical comparison across hashed tags in said at least partial structural representation of the document to thereby identify said first hashed start and end tags and said second hashed start and end tags; and verifying that the second hashed start and end tags indicate a proper nesting in relation to said first hashed start

and end tags as taught by Girardot, because Hind, and Girardot are from the same field of endeavor of providing a method, and system which utilizing XML Syntactical properties to compress and validating XML element data, and to minimizing the size of files being transmitted and stored (Hind col 3, lines 5-30).

Regarding claim 22, Hind does not expressly teach, but Girardot teaches a method, comprising a further step of: **wherein the numerical comparison is followed by a further step of: performing a string comparison, in accordance with said the syntactic rule, across non-processed parts of respective tags in the encoded representation of the document.** For example, Girardot discloses XML Validating parser to ensure that the documents conformed to the language (Girardot col. 4, lines 60-65). Furthermore Girardot discloses a SAX event 270 for each received element. Possibly, the system can also build dynamically a DOM tree 280. Each time an element is received, it is added dynamically to the DOM tree (Girardot col. 7, lined 25-30, Fig. 2). Also Girardot discloses hereinbelow some exemplary code of the present invention, wherein system to fill in this element code space. That is, for each element declaration, the system gets the element name from the array on the client side, and the hash table on the server side, and the required and fixed attributes from the array on the client side, and the hash table on the server side (Girardot, col 10, lines 40-35): `*@param dtd a <code>DTD</code> object */`. The examiner interprets the claimed **start and end tags** as equivalent to Girardot's teaching of hash table is used on the server for a specific tag name or attribute name or value wherein server code spaces is performed by the server filling in the hash table for an element code space, for example `*@param dtd` is parameter element to a specific element type, `<code>DTD</code>` is start tag and end tag in code instead of the input string of a hash table, and because the Applicant

Specification defines, “ this idea allows an arbitrary XML tag to be treated as a numeral or code” (Applicant specification page 7 para 207), and the claimed **the syntactic rule relates to proper nesting of tags** as equivalent to Girardot’s teaching of XML Validating parser to ensure that the documents conformed to the language of SAX event, and because and because the Applicant Specification defines, “FIGS. 2(a) and 2(b) depict a prior art SAX parser process 236, which supports optional well-formedness and/or validation checking sub-processes” (Applicant specification page 7 para 209, fig. 2a-b).

It would have been obvious to one of ordinary skill in the art at the time the invention to modify the teaching of Hind to include wherein the numerical comparison is followed by a further step of: performing a string comparison, in accordance with said the syntactic rule, across non-processed parts of respective tags in the encoded representation of the document as taught by Girardot, because Hind, and Girardot are from the same field of endeavor of providing a method, and system which utilizing XML Syntactical properties to compress and validating XML element data, and to minimizing the size of files being transmitted and stored (Hind col 3, lines 5-30).

Regarding claim 23, Hind does not expressly teach, but Girardot teaches a method, comprising a further step of: **checking the well-formedness of the encoded representation of the document against a syntactic rule**. For example, Girardot discloses XML Validating parser to ensure that the documents conformed to the language (Girardot col. 4, lines 60-65). The Examiner reads the claimed **the syntactic rule** as equivalent to Girardot’s teaching of XML Validating parser to ensure that the documents conformed to the language of SAX event, and because and because the Applicant Specification defines, “FIGS. 2(a) and 2(b) depict a prior art

SAX parser process 236, which supports optional well-formedness and/or validation checking sub-processes” (Applicant specification page 7 para 209, fig. 2a-b).

It would have been obvious to one of ordinary skill in the art at the time the invention to modify the teaching of Hind to include checking the well-formedness of the encoded representation of the document against a syntactic rule as taught by Girardot, because Hind, and Girardot are from the same field of endeavor of providing a method, and system which utilizing XML Syntactical properties to compress and validating XML element data, and to minimizing the size of files being transmitted and stored (Hind col 3, lines 5-30).

Regarding claim 24 Hind does not expressly teach, but Girardot teaches a method, comprising a further step of: **wherein the syntactic rule relates to proper nesting of tags and said checking step comprises sub-steps of: performing a numerical comparison across hashed tags in the encoded representation of the document, thereby to identify the first hashed start and end tags and said the extended hashed start and end tags; and verifying that the extended hashed start and end tags indicate a proper nesting in relation to the first hashed start and end tags.** For example, Girardot discloses XML Validating parser to ensure that the documents conformed to the language (Girardot col. 4, lines 60-65). Furthermore Girardot discloses a SAX event 270 for each received element. Possibly, the system can also build dynamically a DOM tree 280. Each time an element is received, it is added dynamically to the DOM tree (Girardot col. 7, lined 25-30, Fig. 2). Also Girardot discloses hereinbelow some exemplary code of the present invention, wherein system to fill in this element code space. That is, for each element declaration, the system gets the element name from the array on the client side, and the hash table on the server side, and the required and fixed attributes from the array on

Art Unit: 2176

the client side, and the hash table on the server side (Girardot, col 10, lines 40-35): `*@param dtd` a `<code>DTD</code>` object `*/`. The examiner interprets the claimed **start and end tags** as equivalent to Girardot's teaching of hash table is used on the server for a specific tag name or attribute name or value wherein server code spaces is performed by the server filling in the hash table for an element code space, for example `*@param dtd` is parameter element to a specific element type, `<code>DTD</code>` is start tag and end tag in code instead of the input string of a hash table, and because the Applicant Specification defines, "this idea allows an arbitrary XML tag to be treated as a numeral or code" (Applicant specification page 7 para 207), and the claimed **the syntactic rule relates to proper nesting of tags** as equivalent to Girardot's teaching of XML Validating parser to ensure that the documents conformed to the language of SAX event, and because and because the Applicant Specification defines, "FIGS. 2(a) and 2(b) depict a prior art SAX parser process 236, which supports optional well-formedness and/or validation checking sub-processes" (Applicant specification page 7 para 209, fig. 2a-b).

It would have been obvious to one of ordinary skill in the art at the time the invention to modify the teaching of Hind to include wherein the syntactic rule relates to proper nesting of tags and said checking step comprises sub-steps of: performing a numerical comparison across hashed tags in the encoded representation of the document, thereby to identify the first hashed start and end tags and said the extended hashed start and end tags; and verifying that the extended hashed start and end tags indicate a proper nesting in relation to the first hashed start and end tags as taught by Girardot, because Hind, and Girardot are from the same field of endeavor of providing a method, and system which utilizing XML Syntactical properties to

Art Unit: 2176

compress and validating XML element data, and to minimizing the size of files being transmitted and stored (Hind col 3, lines 5-30).

Regarding claim 25, Hind teaches wherein the numerical comparison is followed by a further step of: performing a string comparison across non-hashed parts of respective tags in the encoded representation of the document. Hind discloses entries in a table of strings where once all the string have been located and the strings that meet the condition have been entered into the string table, the logic processes this string table, using it to perform the substitution of entries for strings within the input file (col 10, line 66 – col 11, line 15).

Regarding claim 26, Hind does not expressly teach, but Girardot teaches, wherein said well-formedness checking step is one of (a) succeeded by, (b) included in, and (c) replaced by a validation step against a validation reference document VRD, said validation step comprising sub-steps of: (a) processing the VRD, said processing comprising, for a syntactic element in the VRD, sub-sub-steps of: (i) identifying a type of the syntactic element of the VRD; and (ii) processing the syntactic element by applying a hash function thereto if said the type is the first type, the hash function generating a numeric code from the element; and (b) checking the encoded representation of the markup language document against the processed VRD, said checking comprising a sub-sub-step of numerically comparing corresponding numeric codes of the elements. For example, Girardot discloses XML Validating parser to ensure that the documents conformed to the language (Girardot col. 4, lines 60-65). Furthermore Girardot discloses a SAX event 270 for each received element. Possibly, the system can also build dynamically a DOM tree 280. Each time an element is received, it is added dynamically to the DOM tree (Girardot col. 7, lined 25-30, Fig. 2). Also

Art Unit: 2176

Girardot discloses hereinbelow some exemplary code of the present invention, wherein system to fill in this element code space. That is, for each element declaration, the system gets the element name from the array on the client side, and the hash table on the server side, and the required and fixed attributes from the array on the client side, and the hash table on the server side (Girardot, col 10, lines 40-35): `*@param dtd a <code>DTD</code> object */`. The examiner interprets the claimed **checking comprising a sub-sub-step** as equivalent to Girardot's teaching of hash table is used on the server for a specific tag name or attribute name or value wherein server code spaces is performed by the server filling in the hash table for an element code space, and the claimed **a validation reference document VRD** as equivalent to Girardot's teaching of XML Validating parser to ensure that the documents conformed to the language of SAX event, and because and because the Applicant Specification defines, "FIGS. 2(a) and 2(b) depict a prior art SAX parser process 236, which supports optional well-formedness and/or validation checking sub-processes" (Applicant specification page 7 para 209, fig. 2a-b).

It would have been obvious to one of ordinary skill in the art at the time the invention to modify the teaching of Hind to include wherein said well-formedness checking step is one of (a) succeeded by, (b) included in, and (c) replaced by a validation step against a validation reference document VRD, said validation step comprising sub-steps of: (a) processing the VRD, said processing comprising, for a syntactic element in the VRD, sub-sub-steps of: (i) identifying a type of the syntactic element of the VRD; and (ii) processing the syntactic element by applying a hash function thereto if said the type is the first type, the hash function generating a numeric code from the element; and (b) checking the encoded representation of the markup language document against the processed VRD, said checking comprising a sub-sub-step of numerically

comparing corresponding numeric codes of the elements as taught by Girardot, because Hind, and Girardot are from the same field of endeavor of providing a method, and system which utilizing XML Syntactical properties to compress and validating XML element data, and to minimizing the size of files being transmitted and stored (Hind col 3, lines 5-30).

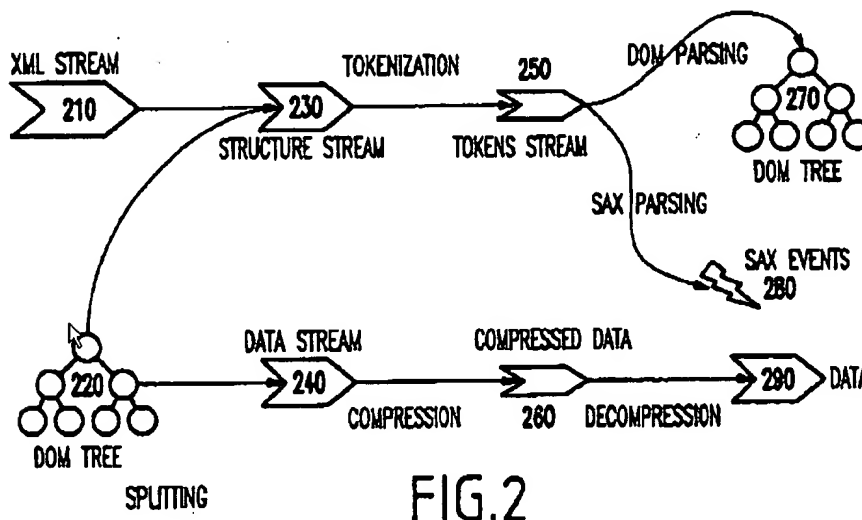
Regarding claim 29, Hind teaches **wherein said numerically comparing step is succeeded by a further step of string-comparing corresponding non-processed representations of elements not of the first type**. For example Hind discloses entries in a table of strings where once all the string have been located and the strings that meet the condition have been entered into the string table, the logic processes this string table, using it to perform the substitution of entries for strings within the input file (Hind col 10, line 66 – col 11, line 15)

Regarding claim 30, Hind teaches **a declaration of said structural element**. Hind discloses declaration of an entity (Hind col 8, lines 59-65, and Hind col 9, lines 20-35 shows an example of an entity declaration).

Regarding claim 31, Hind teaches **wherein said structural element is a tag**. Hind discloses tag syntax for the entity declaration (Hind col 8, lines 57-62).

Regarding claim 59, Hind does not teach, but Girardot teach **the step of validating the markup language document against a validation reference document (VRD)**. For example, Girardot discloses XML Validating parser to ensure that the documents conformed to the language (Girardot col. 4, lines 60-65). Furthermore Girardot discloses a SAX event 270 for each received element. Possibly, the system can also build dynamically a DOM tree 280. Each time an element is received, it is added dynamically to the DOM tree (Girardot col. 7, lined 25-30, Fig. 2). The examiner interprets the claimed **a validation reference document (VRD)** as equivalent

to Girardot's teaching of SAX event 270, and because and because the Applicant Specification defines, "FIGS. 2(a) and 2(b) depict a prior art SAX parser process 236, which supports optional well-formedness and/or validation checking sub-processes" (Applicant specification page 7 para 209, fig. 2a-b).



In addition Girardot teaches (a) **processing the markup language document, for each document tag identified therein, if the document tag is not a first document tag in a corresponding markup language document tag hierarchy, said processing comprising the sub-sub-steps (i) determining a hierarchy position of the document tag.** For example, Girardot discloses XML Validating parser to ensure that the documents conformed to the language (Girardot col. 4, lines 60-65). Furthermore Girardot discloses a SAX event 270 for each received element. Possibly, the system can also build dynamically a DOM tree 280. Each time an element is received, it is added dynamically to the DOM tree (Girardot col. 7, lined 25-30, Fig. 2

Furthermore Girardot teaches (ii) **determining an extended numeric code of the document tag concatenated with a numeric code of a previous document tag in the**

Art Unit: 2176

document tag hierarchy; and (iii) storing the extended numeric code of the document tag if the document tag is more deeply nested than a previous document tag, (ii) determining an extended numeric code of the tag concatenated with a numeric code of a previous tag in the corresponding tag hierarchy. Specifically Girardot discloses hereinbelow some exemplary code of the present invention, wherein system to fill in this element code space. That is, for each element declaration, the system gets the element name from the array on the client side, and the hash table on the server side, and the required and fixed attributes from the array on the client side, and the hash table on the server side (Girardot, col 10, lines 40-35): `*@param dtd a <code>DTD</code> object */`. The examiner interprets the claimed **concatenating the first hashed start tag with the second hashed start tag** as equivalent to Girardot's teaching of hash table is used on the server for a specific tag name or attribute name or value wherein server code spaces is performed by the server filling in the hash table for an element code space, for example `*@param dtd` is parameter element to a specific element type, `<code>DTD</code>` is start tag and end tag in code instead of the input string of a hash table, and because the Applicant Specification defines, "this idea allows an arbitrary XML tag to be treated as a numeral or code" (Applicant specification page 7 para 207).

In addition Girardot teaches **(b) processing the VRD, for each tag identified therein, if the tag is not a first tag in a corresponding tag hierarchy, said processing comprising sub-steps of: (i) determining a hierarchy position of the tag.** For example, Girardot discloses XML Validating parser to ensure that the documents conformed to the language (Girardot col. 4, lines 60-65). Furthermore Girardot discloses a SAX event 270 for each received element. Possibly, the system can also build dynamically a DOM tree 280. Each time an element is

received, it is added dynamically to the DOM tree (Girardot col. 7, lined 25-30, Fig. 2).

Furthermore Girardot discloses the attribute code space, for each element declared, the server system obtains the corresponding attribute declaration from the previously built DOM tree (step 440). It adds the attribute name in the hash table with the attribute name as the key and (256.times.pageNumber+index) as the value (step 445) (Girardot col. 8, lines 25-35, fig. 4)

In addition Girardot teaches **(iii) storing the extended numeric code of the tag in a list.** Specifically Girardot discloses hash table (Girardot, col 10, lines 40-35)

Also Girardot teaches **(c) validating the markup language document if the extended numeric code of the document tag is one of found in the list and is a valid subset of a member of the list.** For example, Girardot discloses XML Validating parser to ensure that the documents conformed to the language (Girardot col. 4, lines 60-65). Furthermore Girardot discloses a SAX event 270 for each received element. Possibly, the system can also build dynamically a DOM tree 280. Each time an element is received, it is added dynamically to the DOM tree (Girardot col. 7, lined 25-30, Fig. 2). The examiner interprets the claimed **a validation reference document (VRD)** as equivalent to Girardot's teaching of SAX event 270, and because and because the Applicant Specification defines, "FIGS. 2(a) and 2(b) depict a prior art SAX parser process 236, which supports optional well-formedness and/or validation checking sub-processes" (Applicant specification page 7 para 209, fig. 2a-b).

It would have been obvious to one of ordinary skill in the art at the time the invention to modify the teaching of Hind to include the step of validating the markup language document against a validation reference document (VRD), and (a) processing the markup language document, for each document tag identified therein, if the document tag is not a first document

Art Unit: 2176

tag in a corresponding markup language document tag hierarchy, said processing comprising the sub-sub-steps (i) determining a hierarchy position of the document tag; (ii) determining an extended numeric code of the document tag concatenated with a numeric code of a previous document tag in the document tag hierarchy; and (iii) storing the extended numeric code of the document tag if the document tag is more deeply nested than a previous document tag; (b) processing the VRD, for each tag identified therein, if the tag is not a first tag in a corresponding tag hierarchy, said processing comprising sub-sub-steps of: (i) determining a hierarchy position of the tag; (ii) determining an extended numeric code of the tag concatenated with a numeric code of a previous tag in the corresponding tag hierarchy; and (iii) storing the extended numeric code of the tag in a list; and (c) validating the markup language document if the extended numeric code of the document tag is one of found in the list and is a valid subset of a member of the list as taught by Girardot, because Hind, and Girardot are from the same field of endeavor of providing a method, and system which utilizing XML Syntactical properties to compress and validating XML element data, and to minimizing the size of files being transmitted and stored (Hind col 3, lines 5-30).

Regarding claim 60, Hind does not teach, but Girardot teach determining a compressed representation of the syntactic element if the type is not the first type. For example, Girardot discloses XML Validating parser to ensure that the documents conformed to the language (Girardot col. 4, lines 60-65). The examiner interprets the claimed **if the type is not the first type** as equivalent to Girardot's teaching of SAX event 270, and because and because the Applicant Specification defines, "FIGS. 2(a) and 2(b) depict a prior art SAX parser process

236, which supports optional well-formedness and/or validation checking sub-processes”
(Applicant specification page 7 para 209, fig. 2a-b).

Regarding claim 61, Hind teaches identifying an encoded representation. For example, Hind discloses a compression of a structured document and a document type definition (Hind, Title), where XML and DTD files are compressed, reducing the file size for storage and transfer (Hind, col 3, lines 62-67). Specifically, an encoded file is read and each of the tags is located in the encoded file. Thereafter, a subprocess substitutes a unique short tag for each of the unique one of the located tags (Hind, col 4, lines 25-30).

In addition, Hind does not teach, but Girardot teach **processing the encoded element by one of: (i) applying an inverse hash function, complementing the hash function of claim 1, to the encoded element, thereby to generate a syntactic element if the encoded element is a numeric code; and (ii) decompressing the encoded element if the encoded element is not a numeric code; and retaining the syntactic element in the document.** Specifically Girardot discloses hereinbelow some exemplary code of the present invention, wherein system to fill in this element code space. That is, for each element declaration, the system gets the element name from the array on the client side, and the hash table on the server side, and the required and fixed attributes from the array on the client side, and the hash table on the server side (Girardot, col 10, lines 40-35). Furthermore Girardot discloses each time it encounters an element or attribute name or value token, it searches its corresponding text string in the code spaces in order to decode the binary structure (Girardot, col 6, lines 45-46). The Examiner reads the claimed **an inverse hash function** as equivalent to Girardot’s teaching of encoded and decode element or attribute name or value token corresponding text string in the code spaces in hash table.

It would have been obvious to one of ordinary skill in the art at the time the invention to modify the teaching of Hind to include processing the encoded element by one of: (i) applying an inverse hash function, complementing the hash function of claim 1, to the encoded element, thereby to generate a syntactic element if the encoded element is a numeric code; and (ii) decompressing the encoded element if the encoded element is not a numeric code; and retaining the syntactic element in the document as taught by Girardot, because Hind, and Girardot are from the same field of endeavor of providing a method, and system which utilizing XML Syntactical properties to compress and validating XML element data, and to minimizing the size of files being transmitted and stored (Hind col 3, lines 5-30).

Regarding claim 62, the rejection of claim 1 is fully incorporated. In addition Hind teaches **wherein each of said means forms a part of an embedded computer**. Specifically Hind discloses a compression of a structured document and a document type definition (Hind, Title), where XML and DTD files are compressed, reducing the file size for storage and transfer (Hind, col 3, lines 62-67). Specifically, an encoded file is read and each of the tags is located in the encoded file. Thereafter, a subprocess substitutes a unique short tag for each of the unique one of the located tags (Hind, col 4, lines 25-30).

Response to Argument

8. Applicant's arguments filed 10/06/2006 have fully consider but they are not persuasive. Applicant argues that the references, in combination, do not teach the claimed limitations and argues mostly the amended portion of the claims (see Remarks pages 17-22). To address theses amendments, the Examiner introduces Girardot reference (see rejection above for details).

Art Unit: 2176

In addition Applicant argues that Hind does not teach **a hash algorithm**. The Examiner disagrees. For example, Hind discloses entries in a table of strings where once all the string have been located and the strings that meet the condition have been entered into the string table, the logic processes this string table, using it to perform the substitution of entries for strings within the input file (col 10, line 66 – col 11, line 15). It is noted that Hind's logic processes of substitution of entries for strings within the input file is inherently perform a hash algorithm as claimed.

Additionally, the Examiner introduces Girardot reference. Specifically Girardot discloses a hash table is used on the server for a specific tag name or attribute name or value (e.g., which are of string type), and then proceed to step 320 parsing a Document Type Definition (DTD) produces a tree where elements and attributes can be accessed by name (Girardot, col 7, lines 45-65, fig. 3b-c).

TAG CODE SPACE		ATTRIBUTE NAME CODE SPACE		ATTRIBUTE NAME CODE SPACE	
TAG NAME	TOKEN	ATTRIBUTE NAME	TOKEN	ATTRIBUTE VALUE	TOKEN
BOOK	5	AUTHOR	5	LITERATURE	5
TITLE	6	GENRE	6	SCIENCE	6
CHAPTER	7	NUMBER	7	HISTORY	7
PICTURE	8	CAPTION	8	CARTOONS	8

HASH TABLE FOR THE CODE SPACES ON THE SERVER SIDE

Also, Applicant argues that **the proposed combination of Hind and Applicant Admitted Prior Art is not believed to be one that one of merely ordinary skill would attempt, nor would there be a reasonable expectation of success if such combination were to be attempted**. The Examiner disagrees. Specifically Hind Hind's logic processes of

Art Unit: 2176

substitution of entries for strings within the input file are inherently perform a hash algorithm as claimed, while the Applicant Admitted Prior Art in FIGS. 2(a) and 2(b) depict a prior art SAX parser process 236, which supports optional well-formedness and/or validation checking sub-processes” (Applicant specification page 7 para 209, fig. 2a-b). It would have been obvious to one of ordinary skill in the art at the time the invention to modify the teaching of Hind to include well-formedness and/or validation checking sub-processes as described by the Applicant Admitted Prior Art, because Hind, and Applicant Admitted Prior Art are from the same field of endeavor of providing a method, and system which utilizing XML Syntactical properties to compress and validating XML element data, and to minimizing file size the size of files being transmitted and stored (Hind col 3, lines 5-30).

However, to clarify the Applicant amendments filed on 10/06/2006, the Examiner replaces the Applicant Admitted Prior Art and Fernandez references with Girardot reference and (see rejection above for details).

Therefore the Examiner respectfully maintains the rejection of claims 1-7, 9-17, 20-26, 29-31, 34, 39, 44, 47-48 and 59-62 at this time.

Conclusion

9. **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after

Art Unit: 2176


the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Quoc A. Tran whose telephone number is 571-272-8664. The examiner can normally be reached on Monday through Friday from 9 AM to 5 PM EST.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Herndon R. Heather can be reached on 571-272-4136. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Quoc A. Tran
Patent Examiner
Technology Center 2176
December 18, 2006


Heather R. Herndon
Supervisory Patent Examiner
Technology Center 2100